CS 134: Networks	Problem set 7
Prof. Yaron Singer	Due: **Thursday 3/30/2017**

Overview

In the independent cascade model, we are given a directed weighted graph $G = (V, E, \mathbf{p})$ where $\mathbf{p}$ is a vector of length $|E|$ that encodes the probabilities of each node infecting its neighbor. Influence in this model corresponds to a stochastic process that mimics an epidemic: in time step $t = 0$ an initial set of nodes is chosen to be infected, then at each time step $t \geq 1$, every node $u \in V$ who became infected at time step $t - 1$ tries to infect every one of its neighbors v and succeeds with probability $p_{u,v}$. This is a stochastic process that terminates after $n = |V|$ steps. The *influence* of a set of nodes S in this model is simply the expected number of nodes that became infected in some time step $t \in \{1, \ldots, n\}$. We write the influence of a set of nodes as a function $f : 2^{|V|} \to \mathbb{R}$:

$$f(S) = \text{the expected number of nodes infected when } S \text{ is chosen as the initial set of adopters}$$

In class, we proved that for every independent cascade model, the corresponding function is *monotone* and *submodular*. We showed this using the following argument. In the independent cascade model, given an initial set of adopters S (i.e. the set S infected at $t = 0$), a node $v \in V \setminus S$ ends up being infected if there is a path of infected nodes between v and S. Putting it differently, v becomes infected if there is a path of *realized* edges from S to v. If E_i is the set of edges that are realized, then the nodes that a set S infects is the set of nodes that are reachable from S in the graph $G_i = (V, E_i)$. We will use $R_i(S)$ to denote this set. Thinking about influence in this way, we can enumerate all possible subsets of edges $E_1, E_2, \ldots, E_m$ and the corresponding graphs $\{G_i = (V, E_i)\}_{i=1}^m$ and compute the probability $\mathbb{P}[G = G_i]$ of each graph realizing. Thus, the expected number of nodes infected by an initial set of adopters is:

$$f(S) = \sum_{i=1}^{m} \mathbb{P}[G = G_i] \cdot |R_i(S)|$$

If we think about each node u in G_i as a "circle" and all the nodes that are reachable from it as "points", then for every $i \in \{1, \ldots, m\}$ we have that $R_i(S)$ is the set of all "points" covered by the set of "circles" in S. Thus, the function $r_i : 2^V \to \mathbb{R}$ defined as $r_i(S) = |R_i(S)|$ is a coverage function, which is monotone and submodular, and since positive weighted sums of monotone submodular functions are themselves monotone submodular, we have that our function f is also monotone and submodular.

Submodularity and NP-Hardness. In influence maximization, our goal this week is to identify a set of k nodes in the network s.t. infecting them at time step $t = 0$ will maximize the expected number of nodes infected in the graph. It turns out that this is an NP-hard problem, so unless P=NP there are no algorithms whose run time is polynomial in the size of the graph that can solve this problem optimally. Fortunately, however, the fact that the influence function is monotone and submodular implies that we can apply the greedy algorithm and find a set of k initial adopters to be infected at $t = 0$ that will, in expectation, infect at least a $1 - 1/e$ fraction of nodes as the optimal set of initial adopters. This is because we proved that for any monotone submodular function the greedy algorithm finds a solution S whose value $f(S)$ is at least a $1 - 1/e$ fraction of the value of the optimal solution.

From theory to practice. As we start thinking about implementation of influence maximization algorithms, we discover that there is still a gap that we need to bridge. Although the influence

function f is submodular, it requires summing over exponentially (in n) many realizations G_i. Thus, even though the greedy algorithm is computationally efficient when the submodular function can be evaluated in polynomial time, evaluating the function is computationally expensive since it requires summing over exponentially-many terms. In this problem set, we show how to overcome this problem using *sampling*. We will first show that for our influence function, we can obtain arbitrarily good approximations of the marginal contribution of a node using a modest number of samples. Then, we will consider a modified version of the greedy algorithm in which we don't know how to evaluate the marginal contribution of elements precisely, but only approximately. We will show that in this modified version, the quality of the solution we obtain gracefully degrades as a function of our approximation of the marginal contributions. Thus, putting these two bounds together we will see that we can run a modified version of the greedy algorithm in which we *sample* the marginal contributions of nodes, and obtain a desirable approximation guarantee.

Beyond influence. When we deal with real data it is quite common that we do not know how to exactly evaluate the objective we are interested in optimizing. Sampling is a powerful technique that is extremely useful in data science and machine learning, and in the future you will likely apply the principles we use here in a broad range of applications.

1. Estimating Marginal Contribution to Influence using Sampling. (30 points)

a. (5 points) For a given set of nodes $S \subseteq V$, let X_S be the random variable that receives value $r_i(S)$ with probability $\mathbb{P}[G = G_i]$. Express the value of the influence function when S is the set of initial adopters, i.e. express $f(S)$ in terms of X_S;

b. (5 points) For a given node $a \in V$ and set of nodes $S \subseteq N$, express the marginal contribution of a to S, i.e. express $f_S(a)$ in terms of $X_{S\cup a}$ and X_S;

c. (15 points) Recall that the empirical mean of ℓ samples is the average value of the ℓ samples. In order to approximate $f_S(a)$ using the means of $X_{S\cup a}$ and X_S, we need to bound the difference between the empirical mean of a random variable and its true mean. To do so, we can use the following version of the famous Chernoff bound:

Theorem. *Let $X_1, X_2, \ldots, X_\ell$ be independent random variables s.t. for every $i \in [1, \ell]$ we have that $\mathbb{E}[X_i] = \mu$ and $X_i \in [0, n]$. Then for any $\delta \geq 0$:*

$$\mathbb{P}\left[\left|\frac{1}{\ell}\sum_{i=1}^{\ell} X_i - \mu\right| \geq \delta\right] \leq 2e^{\frac{-2\ell\delta^2}{n^2}}$$

For a set S and node a, use the Chernoff bound to calculate the number of samples ℓ required to approximate $\mathbb{E}[X_{S\cup a}]$ and $\mathbb{E}[X_S]$ so that you can compute an estimate $\tilde{f}_S(a)$ of $f_S(a)$ s.t. with probability at least $1 - \frac{1}{n^2 k}$:

$$|\tilde{f}_S(a) - f_S(a)| \leq \epsilon$$

d. (5 points) Describe an algorithm that given a precision parameter $\epsilon > 0$ and k, for any $S \subseteq V$ the algorithm returns an element $a \notin S$ s.t. with probability at least $1 - \frac{1}{n \cdot k}$ we have:

$$f_S(a) \geq \max_{b \in V} f_S(b) - \frac{\epsilon^2}{k}$$

i.e. the algorithm returns an element whose marginal contribution is within $\frac{\epsilon^2}{k}$ of the true maximum marginal contribution of elements not in S. *Hint:* It might be useful to recall the union bound: for any countable set of events $A_1, A_2, \ldots$, the following holds: $\mathbb{P}\left(\bigcup_i A_i\right) \leq \sum_i \mathbb{P}(A_i)$.

2. The greedy algorithm with approximate marginals. (30 points)

a. (10 points) For some submodular function $f : 2^V \to \mathbb{R}$, let $S \subseteq V$, and $O \subseteq V$. Let $o^\star \in \arg\max_{o \in O} f_S(o)$ and assume that for every $o \notin S$ we have that $f_S(o) \geq \frac{\epsilon}{k}$, for some $\epsilon > 0$ and k. Suppose that we have an element $a \in V$ that respects $f_S(a) \geq \max_{b \in V} f_S(b) - \frac{\epsilon^2}{k}$. Show that:

$$f_S(a) \geq (1 - \epsilon) f_S(o^\star)$$

b. (20 points) Let O be an optimal solution for maximizing a submodular function $f : 2^V \to \mathbb{R}$ under cardinality constraint k, i.e. O is a set of elements of size k for which $f(O) = \max_{T:|T| \leq k} f(T)$. Use the analysis technique we used in class to show that a modified version of the greedy algorithm in which in every iteration the algorithm selects an element a which respects $f_S(a) \geq (1 - \epsilon) f_S(o^\star)$ where S is the set of elements selected by the greedy algorithm in all previous iterations and $o^\star \in \arg\max_{o \in O} f_S(o)$ returns a solution S s.t.:

$$f(S) \geq (1 - 1/e^{1-\epsilon})\texttt{OPT}$$

3. Putting it all together. (20 points) Given a weighted graph $G = (V, E, \mathbf{p})$ let $f : 2^V \to \mathbb{R}$ be the expected number of nodes who were infected in some time step $t \in \{1, \ldots, |V|\}$ according to the independent cascade model, when selecting a set S as the initial set of adopters at time step $t = 0$. Let $n = |V|$. Describe a modification of the greedy algorithm for maximizing influence, which given $k \leq n$ and precision parameter $\epsilon > 0$, a graph $G = (V, E, \mathbf{p})$ returns a set S of k nodes, s.t. with probability $1 - 1/n$ respects:

$$f(S) \geq (1 - 1/e - \epsilon)\texttt{OPT}$$

To make analysis simpler, you may assume that $\texttt{OPT} \geq 1$ and that the marginal contributions never drops below ϵ/k (if you like you can tell us why this is without loss of generality).

4. Programming: Maximizing Influence on Networks (20 points) (Note: for this problem, you will have to use the following algorithm) Given a set of nodes S, consider the following algorithm for sampling S's influence in the Independent Cascade model, an approximation for $f(S)$:

Algorithm 1 SampleInfluence

Graph $G = (V, E)$, with edge probabilities $\{p_{v,w}\}_{(v,w) \in E}$. Subset $S \subseteq G$, sample limit $m \in \mathbb{N}$
for i = 1 **to** m **do**
    Realize every edge in $(v, w) \in E$ with probability $p_{v,w}$ and set E' to be the set of realized edges.
    Set r_i to be the number of nodes in V reachable from any $s \in S$ (via, say, a BFS search).
end for
Set $f(S) = \frac{1}{m} \sum_{i=1}^{m} r_i$
return $f(S)$

In effect, we repeatedly sample realizations of the overall graph to see how many nodes S can realize, then take the average.

Raynor's finished answering Piazza questions for the night, and now he's already planning for Datamatch 2018. He'd like to generate some publicity by offering some lovebirds totally paid meals at L'Espalier, but L'Espalier is expensive and Raynor is poor, so he can only offer meals to 5 students. Furthermore, he wants the people who actually get the meals to virally advertise for Datamatch, so he'd like to pick the students with the most :fire: Snapchat game so that they can convince the most other students to try Datamatch. How should Raynor approach picking these students?

The file 'network.txt' is a modified version of a real voting network available on SNAP. The file is a list of directed edges with weights, formatted such that $a \quad b \quad c$ indicates an edge from node a to node b with edge weight c such that node a, when activated, succeeds in activating node b with probability c as described in the Independent Cascade model. (Imagine student a's Snapchat game only convinces student b to try Datamatch with success c. Emojis tend to increase this weight.)

a. **(1 point)** First, load your network file. What is the probability of node 42 influencing node 75?

b. **(4 points)** Write a function $genRandGraph(weights)$ that takes a graph $weights$ with weighted, directed edges, and returns a randomly realized graph as described in the subroutine of SampleInfluence. Run $genRandGraph$ 100 times. What is the average number of nodes in the randomly realized graph?

c. **(7 points)** Write a function $sampleInfluence(G, S, m)$ that takes a weighted graph G, a subset of nodes S, a sample limit m, and returns the influence $f(S)$ of S according to the SampleInfluence algorithm. Run $sampleInfluence(G, S, m)$, where G is your network from a), $S = [17, 23, 42, 2017]$, and $m = 500$. What is $f(S)$?

d. **(8 points)** Using the Greedy Algorithm described in problem 3, and SampleInfluence to approximate influence, what 5 students (nodes) should Raynor pick to give free L'Espalier to? For this subpart, please use a value of $m \geq 10$. (No, you cannot just give it to yourself.) In addition, please report the value of $f_S(a)$ at every step as you build your subset S, as described in the Greedy Algorithm. Finally, please give a short description of your code as a whole, including $genRandGraph$ and $sampleInfluence$, and additionally submit your code as a separate file.

Hint: note that the final run of your Greedy Algorithm may take around an hour. It's strongly encouraged to debug as you go, and to not leave this code until the last moment!